

Latent Tree Language Model

Tomáš Bryhcín

NTIS – New Technologies for the Information Society,
Faculty of Applied Sciences, University of West Bohemia,
Technická 8, 306 14 Plzeň, Czech Republic
bryhcín@kiv.zcu.cz
nlp.kiv.zcu.cz

Abstract

In this paper we introduce Latent Tree Language Model (LTLM), a novel approach to language modeling that encodes syntax and semantics of a given sentence as a tree of word roles.

The learning phase iteratively updates the trees by moving nodes according to Gibbs sampling. We introduce two algorithms to infer a tree for a given sentence. The first one is based on Gibbs sampling. It is fast, but does not guarantee to find the most probable tree. The second one is based on dynamic programming. It is slower, but guarantees to find the most probable tree. We provide comparison of both algorithms.

We combine LTLM with 4-gram Modified Kneser-Ney language model via linear interpolation. Our experiments with English and Czech corpora show significant perplexity reductions (up to 46% for English and 49% for Czech) compared with standalone 4-gram Modified Kneser-Ney language model.

1 Introduction

Language modeling is one of the core disciplines in natural language processing (NLP). Automatic speech recognition, machine translation, optical character recognition, and other tasks strongly depend on the language model (LM). An improvement in language modeling often leads to better performance of the whole task. The goal of language modeling is to determine the joint probability of a sentence. Currently, the dominant approach is n-gram language modeling, which decomposes

the joint probability into the product of conditional probabilities by using the *chain rule*. In traditional n-gram LMs the words are represented as distinct symbols. This leads to an enormous number of word combinations.

In the last years many researchers have tried to capture words contextual meaning and incorporate it into the LMs. Word sequences that have never been seen before receive high probability when they are made of words that are semantically similar to words forming sentences seen in training data. This ability can increase the LM performance because it reduces the *data sparsity* problem. In NLP a very common paradigm for word meaning representation is the use of the *Distributional hypothesis*. It suggests that two words are expected to be semantically similar if they occur in similar contexts (they are similarly distributed in the text) (Harris, 1954). Models based on this assumption are denoted as distributional semantic models (DSMs).

Recently, semantically motivated LMs have begun to surpass the ordinary n-gram LMs. The most commonly used architectures are *neural network LMs* (Bengio et al., 2003; Mikolov et al., 2010; Mikolov et al., 2011) and *class-based LMs*. Class-based LMs are more related to this work thus we investigate them deeper.

Brown et al. (1992) introduced class-based LMs of English. Their unsupervised algorithm searches classes consisting of words that are most probable in the given context (one word window in both directions). However, the computational complexity of this algorithm is very high. This approach was later extended by (Martin et al., 1998; Whit-

taker and Woodland, 2003) to improve the complexity and to work with wider context. Deschacht et al. (2012) used the same idea and introduced Latent Words Language Model (LWLM), where word classes are latent variables in a graphical model. They apply Gibbs sampling or the expectation maximization algorithm to discover the word classes that are most probable in the context of surrounding word classes. A similar approach was presented in (Brychcín and Konopík, 2014; Brychcín and Konopík, 2015), where the word clusters derived from various semantic spaces were used to improve LMs.

In above mentioned approaches, the meaning of a word is inferred from the surrounding words independently of their relation. An alternative approach is to derive contexts based on the syntactic relations the word participates in. Such syntactic contexts are automatically produced by dependency parse-trees. Resulting word representations are usually less topical and exhibit more functional similarity (they are more syntactically oriented) as shown in (Padó and Lapata, 2007; Levy and Goldberg, 2014).

Dependency-based methods for syntactic parsing have become increasingly popular in NLP in the last years (Kübler et al., 2009). Popel and Mareček (2010) showed that these methods are promising direction of improving LMs. Recently, unsupervised algorithms for dependency parsing appeared in (Headden III et al., 2009; Cohen et al., 2009; Spitkovsky et al., 2010; Spitkovsky et al., 2011; Mareček and Straka, 2013) offering new possibilities even for poorly-resourced languages.

In this work we introduce a new DSM that uses tree-based context to create word roles. The word role contains the words that are similarly distributed over similar tree-based contexts. The word role encodes the semantic and syntactic properties of a word. We do not rely on parse trees as a prior knowledge, but we jointly learn the tree structures and word roles. Our model is a soft clustering, i.e. one word may be present in several roles. Thus it is theoretically able to capture the word polysemy. The learned structure is used as a LM, where each word role is conditioned on its parent role. We present the unsupervised algorithm that discovers the tree structures only from the distribution of words in a training corpus (i.e. no labeled data or external sources of in-

formation are needed). In our work we were inspired by class-based LMs (Deschacht et al., 2012), unsupervised dependency parsing (Mareček and Straka, 2013), and tree-based DSMs (Levy and Goldberg, 2014).

This paper is organized as follows. We start with the definition of our model (Section 2). The process of learning the hidden sentence structures is explained in Section 3. We introduce two algorithms for searching the most probable tree for a given sentence (Section 4). The experimental results on English and Czech corpora are presented in Section 6. We conclude in Section 7 and offer some directions for future work.

2 Latent Tree Language Model

In this section we describe Latent Tree Language Model (LTLM). LTLM is a generative statistical model that discovers the tree structures hidden in the text corpus.

Let \mathbf{L} be a word vocabulary with total of $|\mathbf{L}|$ distinct words. Assume we have a training corpus \mathbf{w} divided into S sentences. The goal of LTLM or other LMs is to estimate the probability of a text $P(\mathbf{w})$. Let N_s denote the number of words in the s -th sentence. The s -th sentence is a sequence of words $\mathbf{w}_s = \{w_{s,i}\}_{i=0}^{N_s}$, where $w_{s,i} \in \mathbf{L}$ is a word at position i in this sentence and $w_{s,0} = < s >$ is an artificial symbol that is added at the beginning of each sentence.

Each sentence s is associated with the dependency graph \mathbf{G}_s . We define the dependency graph as a labeled directed graph, where nodes correspond to the words in the sentence and there is a label for each node that we call *role*. Formally, it is a triple $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s, \mathbf{r}_s)$ consisting of:

- The set of nodes $\mathbf{V}_s = \{0, 1, \dots, N_s\}$. Each token $w_{s,i}$ is associated with node $i \in \mathbf{V}_s$.
- The set of edges $\mathbf{E}_s \subseteq \mathbf{V}_s \times \mathbf{V}_s$.
- The sequence of roles $\mathbf{r}_s = \{r_{s,i}\}_{i=0}^{N_s}$, where $1 \leq r_{s,i} \leq K$ for $i \in \mathbf{V}_s$. K is the number of roles.

The artificial word $w_{s,0} = < s >$ at the beginning of the sentence has always role 1 ($r_{s,0} = 1$). Analogously to \mathbf{w} , the sequence of all \mathbf{r}_s is denoted as \mathbf{r} and sequence of all \mathbf{G}_s as \mathbf{G} .

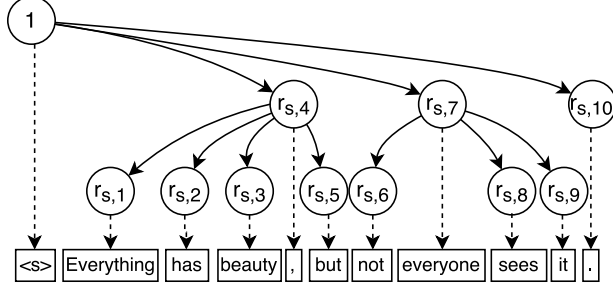


Figure 1: Example of LTLM for the sentence "Everything has beauty, but not everyone sees it."

Edge $e \in \mathbf{E}_s$ is an ordered pair of nodes (i, j) . We say that i is the *head* or the *parent* and j is the *dependent* or the *child*. We use the notation $i \rightarrow j$ for such edge. The directed path from node i to node j is denoted as $i \xrightarrow{*} j$.

We place a few constraints on the graph \mathbf{G}_s .

- The graph \mathbf{G}_s is a *tree*. It means it is the acyclic graph (if $i \rightarrow j$ then not $j \xrightarrow{*} i$), where each node has one parent (if $i \rightarrow j$ then not $k \rightarrow j$ for every $k \neq i$).
- The graph \mathbf{G}_s is *projective* (there are no cross edges). For each edge (i, j) and for each k between i and j (i.e. $i < k < j$ or $i > k > j$) there must exist the directed path $i \xrightarrow{*} k$.
- The graph \mathbf{G}_s is always rooted in the node 0.

We denote these graphs as the *projective dependency trees*. Example of such a tree is on Figure 1. For the tree \mathbf{G}_s we define a function

$$h_s(j) = i, \quad \text{when } (i, j) \in \mathbf{E}_s \quad (1)$$

that returns the parent for each node except the root.

We use graph \mathbf{G}_s as a representation of the *Bayesian network* with random variables \mathbf{E}_s and \mathbf{r}_s . The roles $r_{s,i}$ represent the node labels and the edges express the dependences between the roles. The conditional probability of the role at position i given its parent role is denoted as $P(r_{s,i} | r_{s,h_s(i)})$. The conditional probability of the word at position i in the sentence given its role $r_{s,i}$ is denoted as $P(w_{s,i} | r_{s,i})$.

We model the distribution over words in the sentence s as the mixture

$$P(\mathbf{w}_s) = P(\mathbf{w}_s | r_{s,0}) = \prod_{i=1}^{N_s} \sum_{k=1}^K P(w_{s,i} | r_{s,i} = k) P(r_{s,i} = k | r_{s,h_s(i)}). \quad (2)$$

The root role is kept fixed for each sentence ($r_{s,0} = 1$) so $P(\mathbf{w}_s) = P(\mathbf{w}_s | r_{s,0})$.

We look at the roles as mixtures over child roles and simultaneously as mixtures over words. We can represent dependency between roles with a set of K multinomial distributions θ over K roles, such that $P(r_{s,i} | r_{s,h_s(i)} = k) = \theta_{r_{s,i}}^{(k)}$. Simultaneously, dependency of words on their roles can be represented as a set of K multinomial distributions ϕ over $|\mathbf{L}|$ words, such that $P(w_{s,i} | r_{s,i} = k) = \phi_{w_{s,i}}^{(k)}$. To make predictions about new sentences, we need to assume a prior distribution on the parameters $\theta^{(k)}$ and $\phi^{(k)}$.

We place a Dirichlet prior D with the vector of K hyper-parameters α on a multinomial distribution $\theta^{(k)} \sim D(\alpha)$ and with the vector of $|\mathbf{L}|$ hyper-parameters β on a multinomial distribution $\phi^{(k)} \sim D(\beta)$. In general, D is not restricted to be Dirichlet distribution. It could be any distribution over discrete children, such as logistic normal. In this paper, we focus only on Dirichlet as a conjugate prior to the multinomial distribution and derive the learning algorithm under this assumption.

The choice of the child role depends only on its parent role, i.e. child roles with the same parent are mutually independent. This property is especially important for the learning algorithm (Section 3) and also for searching the most probable trees (Section 4). We do not place any assumption on the length of the sentence N_s or on how many children the parent node is expected to have.

3 Parameter Estimation

In this section we present the learning algorithm for LTLM. The goal is to estimate θ and ϕ in a way that maximizes the predictive ability of the model (generates the corpus with maximal joint probability $P(\mathbf{w})$).

Let $\chi_{(i,j)}^k$ be an operation that changes the tree \mathbf{G}_s to \mathbf{G}'_s

$$\chi_{(i,j)}^k : \mathbf{G}_s \rightarrow \mathbf{G}'_s, \quad (3)$$

such that the newly created tree $G'(V'_s, E'_s, r'_s)$ consists of:

- $V'_s = V_s$.
- $E'_s = (E_s \setminus \{(h_s(i), i)\}) \cup \{(j, i)\}$.
- $r'_{s,a} = \begin{cases} r_{s,a} & \text{for } a \neq i \\ k & \text{for } a = i \end{cases}$, where $0 \leq a \leq N_s$.

It means that we change the role of the selected node i so that $r_{s,i} = k$ and simultaneously we change the parent of this node to be j . We call this operation a *partial change*.

The newly created graph G' must satisfy all conditions presented in Section 2, i.e. it is a projective dependency tree rooted in the node 0. Thus not all partial changes $\chi_{(i,j)}^k$ are possible to perform on graph G_s .

Clearly, for the sentence s there is at most $\frac{N_s(1+N_s)}{2}$ parent changes¹.

To estimate the parameters of LTLTM we apply Gibbs sampling and gradually sample $\chi_{(i,j)}^k$ for trees G_s . For doing so we need to determine the posterior predictive distribution²

$$G'_s \sim P(\chi_{(i,j)}^k(G_s) | w, G), \quad (4)$$

from which we will sample partial changes to update the trees. In the equation, G denote the sequence of all trees for given sentences w and G'_s is a result of one sampling. In the following text we derive this equation under assumptions from Section 2.

The posterior predictive distribution of Dirichlet multinomial has the form of additive smoothing that is well known in the context of language modeling. The hyper-parameters of Dirichlet prior determine how much is the predictive distribution smoothed. Thus the predictive distribution for the word-in-role distribution can be expressed as

$$P(w_{s,i} | r_{s,i}, w_{\setminus s,i}, r_{\setminus s,i}) = \frac{n_{\setminus s,i}^{(w_{s,i} | r_{s,i})} + \beta}{n_{\setminus s,i}^{(\bullet | r_{s,i})} + |L| \beta}, \quad (5)$$

¹The most parent changes are possible for the special case of the tree, where each node i has parent $i - 1$. Thus for each node i we can change its parent to any node $j < i$ and keep the projectivity of the tree. That is $\frac{N_s(1+N_s)}{2}$ possibilities.

²The posterior predictive distribution is the distribution of an unobserved variable conditioned by the observed data, i.e. $P(X_{n+1} | X_1, \dots, X_n)$, where X_i are i.i.d. (independent and identically distributed random variables).

where $n_{\setminus s,i}^{(w_{s,i} | r_{s,i})}$ is the number of times the role $r_{s,i}$ has been assigned to the word $w_{s,i}$, excluding the position i in the s -th sentence. The symbol \bullet represents any word in the vocabulary so that $n_{\setminus s,i}^{(\bullet | r_{s,i})} = \sum_{l \in L} n_{\setminus s,i}^{(l | r_{s,i})}$. We use the symmetric Dirichlet distribution for the word-in-role probabilities as it could be difficult to estimate the vector of hyper-parameters β for large word vocabulary. In the above mentioned equation, β is a scalar.

The predictive distribution for the role-by-role distribution is

$$P(r_{s,i} | r_{s,h_s(i)}, r_{\setminus s,i}) = \frac{n_{\setminus s,i}^{(r_{s,i} | r_{s,h_s(i)})} + \alpha_{r_{s,i}}}{n_{\setminus s,i}^{(\bullet | r_{s,h_s(i)})} + \sum_{k=1}^K \alpha_k}. \quad (6)$$

Analogously to the previous equation, $n_{\setminus s,i}^{(r_{s,i} | r_{s,h_s(i)})}$ denote the number of times the role $r_{s,i}$ has the parent role $r_{s,h_s(i)}$, excluding the position i in the s -th sentence. The symbol \bullet represents any possible role to make the probability distribution summing up to 1. We assume an asymmetric Dirichlet distribution.

We can use predictive distributions of above mentioned Dirichlet multinomials to express the joint probability that the role at position i is k ($r_{s,i} = k$) with parent at position j conditioned on current values of all variables, except those in position i in the sentence s

$$\begin{aligned} P(r_{s,i} = k, j | w, r_{\setminus s,i}) &\propto \\ &P(w_{s,i} | r_{s,i} = k, w_{\setminus s,i}, r_{\setminus s,i}) \\ &\times P(r_{s,i} = k | r_{s,j}, r_{\setminus s,i}) \\ &\times \prod_{a: h_s(a)=i} P(r_{s,a} | r_{s,i} = k, r_{\setminus s,i}). \end{aligned} \quad (7)$$

The choice of the node i role affects the word that is produced by this role and also all the child roles of the node i . Simultaneously, the role of the node i depends on its parent j role. Formula 7 is derived from the joint probability of a sentence s and a tree G_s , where all probabilities which do not depend on the choice of the role at position i are removed and equality is replaced by proportionality (\propto).

We express the final predictive distribution for sampling partial changes $\chi_{(i,j)}^k$ as

$$P(\chi_{(i,j)}^k(\mathbf{G}_s) | \mathbf{w}, \mathbf{G}) \propto \frac{P(r_{s,i} = k, j | \mathbf{w}, \mathbf{r}_{\setminus s,i})}{P(r_{s,i}, h_s(i) | \mathbf{w}, \mathbf{r}_{\setminus s,i})} \quad (8)$$

that is essentially the fraction between the joint probability of $r_{s,i}$ and its parent after the partial change and before the partial change (conditioned on all other variables). This fraction can be interpreted as the necessity to perform this partial change.

We investigate two strategies of sampling partial changes:

- **Per sentence:** We sample a single partial change according to Equation 8 for each sentence in the training corpus. It means during one pass through the corpus (one iteration) we perform S partial changes.
- **Per position:** We sample a partial change for each position in each sentence. We perform in total $N = \sum_{s=1}^S N_s$ partial changes during one pass. Note that the denominator in Equation 8 is constant for this strategy and can be removed.

We compare both training strategies in Section 6. After enough training iterations, we can estimate the conditional probabilities $\phi_l^{(k)}$ and $\theta_k^{(p)}$ from actual samples as

$$\phi_l^{(k)} \approx \frac{n(w_{s,i}=l | r_{s,i}=k) + \beta}{n(\bullet | r_{s,i}=k) + |\mathbf{L}| \beta} \quad (9)$$

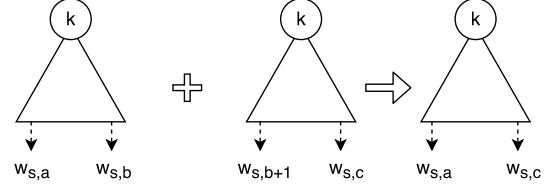
$$\theta_k^{(p)} \approx \frac{n(r_{s,i}=k | r_{s,h_s(i)}=p) + \alpha_k}{n(\bullet | r_{s,h_s(i)}=p) + \sum_{m=1}^K \alpha_m}. \quad (10)$$

These equations are similar to equations 5 and 6, but here the counts n do not exclude any position in a corpus.

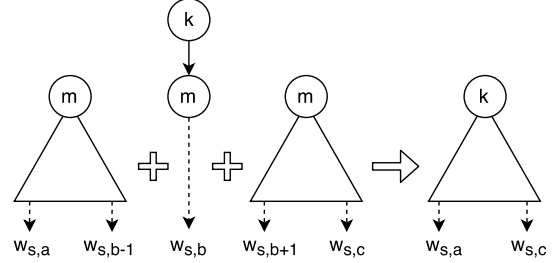
Note that in the Gibbs sampling equation, we assume that the Dirichlet parameters α and β are given. We use a fixed point iteration technique described in (Minka, 2003) to estimate them.

4 Inference

In this section we present two approaches for searching the most probable tree for a given sentence assuming we have already estimated the parameters θ and ϕ .



(a) The root has two or more children.



(b) The root has only one child.

Figure 2: Searching the most probable subtrees.

4.1 Non-deterministic Inference

We use the same sampling technique as for estimating parameters (Equation 8), i.e. we iteratively sample the partial changes $\chi_{(i,j)}^k$. However, we use equations 9 and 10 for predictive distributions of Dirichlet multinomials instead of 5 and 6. In fact, these equations correspond to the predictive distributions over the newly added word $w_{s,i}$ with the role $r_{s,i}$ into the corpus, conditioned on \mathbf{w} and \mathbf{r} . This sampling technique rarely finds the best solution, but often it is very near.

4.2 Deterministic Inference

Here we present the deterministic algorithm that guarantees to find the most probable tree for a given sentence. We were inspired by Cocke-Younger-Kasami (CYK) algorithm (Lange and Leiß, 2009).

Let $\mathbf{T}_{s,a,c}^n$ denote the subtree of \mathbf{G}_s (subgraph of \mathbf{G}_s that is also a tree) containing subsequence of nodes $\{a, a+1, \dots, c\}$. The superscript n denotes the number of children the root of this subtree has. We denote the joint probability of a subtree from position a to position c with the corresponding words conditioned by the root role k as $P^n(\{w_{s,i}\}_{i=a}^c, \mathbf{T}_{s,a,c}^n | k)$. Our goal is to find the tree $\mathbf{G}_s = \mathbf{T}_{s,0,N_s}^{1+}$ that maximizes probability $P(\mathbf{w}_s, \mathbf{G}_s) = P^{1+}(\{w_{s,i}\}_{i=0}^{N_s}, \mathbf{T}_{s,0,N_s}^{1+} | 0)$.

Similarly to CYK algorithm, our approach fol-

lows bottom-up direction and goes through all possible subsequences for a sentence (sequence of words). At the beginning, the probabilities for subsequences of length 1 (i.e. single words) are calculated as $P^{1+}(\{w_{s,a}\}, \mathbf{T}_{s,a,a}^{1+}|k) = P(w_{s,a}|r_{s,a} = k)$. Once it has considered subsequences of length 1, it goes on to subsequences of length 2, and so on.

Thanks to mutual independence of roles under the same parent, we can find the most probable subtree with the root role k and with at least two root children according to

$$P^{2+}(\{w_{s,i}\}_{i=a}^c, \mathbf{T}_{s,a,c}^{2+}|k) = \max_{b:a < b < c} [P^{1+}(\{w_{s,i}\}_{i=a}^b, \mathbf{T}_{s,a,b}^{1+}|k) \times P^{1+}(\{w_{s,i}\}_{i=b+1}^c, \mathbf{T}_{s,b+1,c}^{1+}|k)]. \quad (11)$$

It means we merge two neighboring subtrees with the same root role k . This is the reason why the new subtree has at least two root children. This formula is visualized on Figure 2a. Unfortunately, this does not cover all subtree cases. We find the most probable tree with only root child as follows

$$P^1(\{w_{s,i}\}_{i=a}^c, \mathbf{T}_{s,a,c}^1|k) = \max_{b,m:a \leq b \leq c, 1 \leq m \leq K} [P(w_{s,b}|r_{s,b} = m) \times P(r_{s,b} = m|k) \times P^{1+}(\{w_{s,i}\}_{i=a}^{b-1}, \mathbf{T}_{s,a,b-1}^{1+}|m) \times P^{1+}(\{w_{s,i}\}_{i=b+1}^c, \mathbf{T}_{s,b+1,c}^{1+}|m)]. \quad (12)$$

This formula is visualized on Figure 2b.

To find the most probable subtree no matter how many children the root has, we need to take the maximum from both mentioned equations $P^{1+} = \max(P^{2+}, P^1)$.

The algorithm has complexity $\mathcal{O}(N_s^3 K^2)$, i.e. it has cubic dependence on the length of the sentence N_s .

5 Side-dependent LTLM

Until now, we presented LTLM in its simplified version. In role-by-role probabilities (role conditioned on its parent role) we did not distinguish whether the role is on the left side or the right side of the parent. However, this position keeps important information about the syntax of words (and their roles).

We assume separate multinomial distributions $\hat{\theta}$ for roles that are on the left and $\check{\theta}$ for roles on the right. Each of them has its own Dirichlet prior with hyper-parameters $\hat{\alpha}$ and $\check{\alpha}$, respectively. The process of estimating LTLM parameters is almost the same. The only difference is that we need to redefine the predictive distribution for the role-by-role distribution (Equation 6) to include only counts of roles on the appropriate side. Also, every time the role-by-role probability is used we need to distinguish sides:

$$P(r_{s,i}|r_{s,h_s(i)}) = \begin{cases} \hat{\theta}_{r_{s,i}}^{(r_{s,h_s(i)})} & \text{for } i < h_s(i) \\ \check{\theta}_{r_{s,i}}^{(r_{s,h_s(i)})} & \text{for } i > h_s(i) \end{cases}. \quad (13)$$

In the following text we always assume the side-dependent LTLM.

6 Experimental Results and Discussion

In this section we present experiments with LTLM on two languages, English (EN) and Czech (CS).

As a training corpus we use CzEng 1.0 (Bojar et al., 2012) of the sentence-parallel Czech-English corpus. We choose this corpus because it contains multiple domains, it is of reasonable length, and it is parallel so we can easily provide comparison between both languages. The corpus is divided into 100 similarly-sized sections. We use parts 0–97 for training, the part 98 as a development set, and the last part 99 for testing.

We have removed all sentences longer than 30 words. The reason was that the complexity of the learning phase and the process of searching most probable trees depends on the length of sentences. It has led to removing approximately a quarter of all sentences. The corpus is available in a tokenized form so the only preprocessing step we use is lower-casing. We keep the vocabulary of 100,000 most frequent words in the corpus for both languages. The less frequent words were replaced by the symbol $\langle \text{unk} \rangle$. Statistics for the final corpora are shown in Table 1.

We measure the quality of LTLM by *perplexity* that is the standard measure used for LMs. Perplexity is a measure of uncertainty. The lower perplexity means the better predictive ability of the LM.

Corpora	Sentences	Tokens	OOV rate
EN train	11,530,604	138,034,779	1.30%
EN develop.	117,735	1,407,210	1.28%
EN test	117,360	1,405,106	1.33%
CS train	11,832,388	133,022,572	3.98%
CS develop.	120,754	1,353,015	4.00%
CS test	120,573	1,357,717	4.03%

Table 1: Corpora statistics. *OOV rate* denotes the out-of-vocabulary rate.

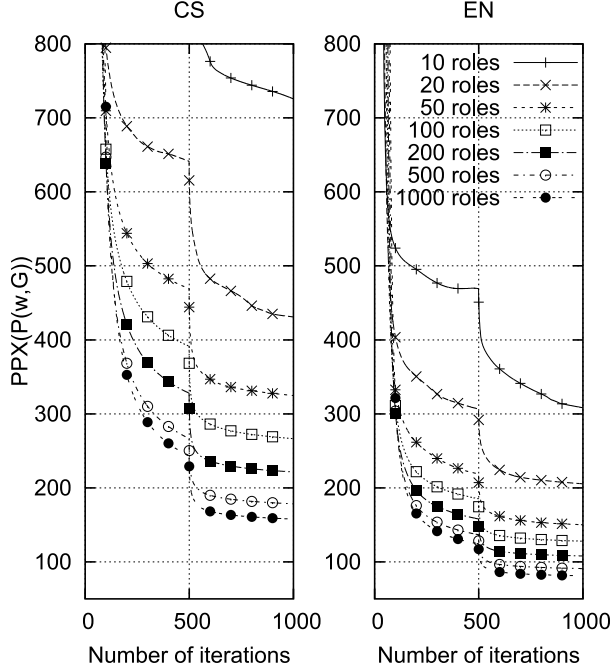


Figure 3: Learning curves of LTLM for both English and Czech. The points in the graphs represent the perplexities in every 100th iteration.

During the process of parameter estimation we measure the perplexity of joint probability of sentences and their trees defined as $PPX(P(w, G)) = \sqrt[N]{\frac{1}{P(w, G)}}$, where N is the number of all words in the training data w .

As we describe in Section 3, there are two approaches for the parameter estimation of LTLM. During our experiments, we found that the per-position strategy of training has the ability to converge faster, but to a worse solution compared to the per-sentence strategy which converges slower, but to a better solution.

We train LTLM by 500 iterations of the per-position sampling followed by another 500 iterations of the per-sentence sampling. This proves to be effi-

Model	EN	CS
2-gram MKN	165.9	272.0
3-gram MKN	67.7	99.3
4-gram MKN	46.2	73.5
300n RNNLM	51.2	69.4
4-gram LWLM	52.7	81.5
PoS STLM	455.7	747.3
1000r STLM	113.7	211.0
1000r det. LTLM	54.2	111.1
4-gram MKN + 300n RNNLM	36.8 (-20.4%)	49.5 (-32.7%)
4-gram MKN + 4-gram LWLM	41.5 (-10.2%)	62.4 (-15.1%)
4-gram MKN + PoS STLM	42.9 (-7.1%)	63.3 (-13.9%)
4-gram MKN + 1000r STLM	33.6 (-27.3%)	50.1 (-31.8%)
4-gram MKN + 1000r det. LTLM	24.9 (-43.1%)	37.2 (-49.4%)

Table 2: Perplexity results on the test data. The numbers in brackets are the relative improvements compared with standalone 4-gram MKN LM.

cient in both aspects, the reasonable speed of convergence and the satisfactory predictive ability of the model. The learning curves are showed on Figure 3. We present the models with 10, 20, 50, 100, 200, 500, and 1000 roles. The higher role cardinality models were not possible to create because of the very high computational requirements. Similarly to the training of LTLM, the non-deterministic inference uses 100 iterations of per-position sampling followed by 100 iterations of per-sentence sampling.

In the following experiments we measure how well LTLM generalizes the learned patterns, i.e. how well it works on the previously unseen data. Again, we measure the perplexity, but of probability $P(w)$ for mutual comparison with different LMs that are based on different architectures ($PPX(P(w)) = \sqrt[N]{\frac{1}{P(w)}}$).

To show the strengths of LTLM we compare it with several state-of-the-art LMs. We experiment with Modified Kneser-Ney (MKN) interpolation (Chen and Goodman, 1998), with Recurrent Neural Network LM (RNNLM) (Mikolov et al., 2010; Mikolov et al., 2011)³, and with LWLM (Deschacht et al., 2012)⁴. We have also created syntactic dependency tree based LM (denoted as STLM). Syntactic dependency trees for both languages are provided within CzEng corpus and are based on

³Implementation is available at <http://rnnlm.org/>. Size of the hidden layer was set to 300 in our experiments. It was computationally intractable to use more neurons.

⁴Implementation is available at <http://liir.cs.kuleuven.be/software.php>.

Model\roles	EN							CS						
	10	20	50	100	200	500	1000	10	20	50	100	200	500	1000
STLM	408.5	335.2	261.7	212.6	178.9	137.8	113.7	992.7	764.2	556.4	451.0	365.9	265.7	211.0
non-det. LTLM	329.5	215.1	160.4	126.5	105.6	86.7	78.4	851.0	536.6	367.4	292.6	235.2	186.1	157.6
det. LTLM	252.4	166.4	115.3	92.0	75.4	60.9	54.2	708.5	390.2	267.8	213.2	167.9	133.5	111.1
4-gram MKN + STLM	42.7	41.6	39.9	37.9	36.3	34.9	33.6	67.5	65.1	61.4	58.3	55.5	52.4	50.1
4-gram MKN + non-det. LTLM	41.1	38.0	35.2	32.7	30.7	28.9	27.8	65.8	59.4	55.1	51.1	47.5	43.7	41.3
4-gram MKN + det. LTLM	39.9	36.4	32.8	30.3	28.1	26.0	24.9	64.4	56.1	51.5	47.3	43.4	39.9	37.2

Table 3: Perplexity results on the test data for LTLMs and STLMs with different number of roles. Deterministic inference is denoted as *det.* and non-deterministic inference as *non-det.*

MST parser (McDonald et al., 2005). We use the same architecture as for LTLM and experiment with two approaches to represent the roles. Firstly, the roles are given by the part-of-speech tag (denoted as PoS STLM). No training is required, all information come from CzEng corpus. Secondly, we learn the roles using the same algorithm as for LTLM. The only difference is that the trees are kept unchanged. Note that both deterministic and non-deterministic inference perform almost the same in this model so we do not distinguish between them.

We combine baseline 4-gram MKN model with other models via linear combination (in the tables denoted by the symbol +) that is simple but very efficient technique to combine LMs. Final probability is then expressed as

$$P(w) = \prod_{s=1}^S \prod_{i=1}^{N_s} [\lambda P^{\text{LM1}} + (\lambda - 1) P^{\text{LM2}}]. \quad (14)$$

In the case of MKN the probability P^{MKN} is the probability of a word $w_{s,i}$ conditioned by 3 previous words with MKN smoothing. For LTLM or STLM this probability is defined as

$$P^{\text{LTLM}}(w_{s,i} | r_{s,h_s(i)}) = \sum_{k=1}^K P(w_{s,i} | r_{s,i} = k) P(r_{s,i} = k | r_{s,h_s(i)}). \quad (15)$$

We use the *expectation maximization* algorithm (Dempster et al., 1977) for the maximum likelihood estimate of λ parameter on the development part of the corpus. The influence of the number of roles on the perplexity is shown in Table 3 and the final

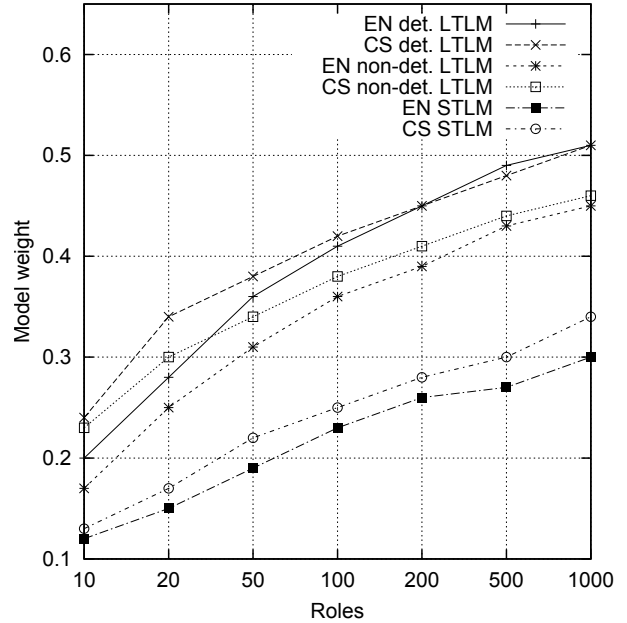


Figure 4: Model weights optimized on development data when interpolated with 4-gram MKN LM.

results are shown in Table 2. Note that these perplexities are not comparable with those on Figure 3 ($\text{PPX}(P(w))$ vs. $\text{PPX}(P(w, G))$). Weights of LTLM and STLM when interpolated with MKN LM are shown on Figure 4.

From the tables we can see several important findings. Standalone LTLM performs worse than MKN on both languages, however their combination leads to dramatic improvements compared with other LMs. Best results are achieved by 4-gram MKN interpolated with 1000 roles LTLM and the deterministic inference. The perplexity was improved by approximately 46% on English and 49% on Czech compared with standalone MKN. The deterministic inference outperformed the non-deterministic one in all cases. LTLM also signifi-

everything	has	beauty	,	but	not	everyone	sees	it	.
it	's	one	,	but	was	he	saw	him	.
that	is	thing	;	course	it	i	made	it	!
let	was	life	—	though	not	she	found	her	...
there	knows	name	-	or	this	they	took	them	'
something	really	father	...	perhaps	that	that	gave	his	what
nothing	says	mother	:	and	the	it	told	me	“
everything	comes	way		maybe	now	who	felt	a	how
here	does	wife	(although	had	you	thought	out	why
someone	gets	place	?	yet	<unk>	someone	knew	that	—
god	has	idea	naught	except	all	which	heard	himself	-

Table 4: Ten most probable word substitutions on each position in the sentence “*Everything has beauty, but not everyone sees it.*” produced by 1000 roles LTLM with the deterministic inference.

cantly outperformed STLM where the syntactic dependency trees were provided as a prior knowledge. The joint learning of syntax and semantics of a sentence proved to be more suitable for predicting the words.

An in-depth analysis of semantic and syntactic properties of LTLM is beyond the scope of this paper. For better insight into the behavior of LTLM, we show the most probable word substitutions for one selected sentence (see Table 4). We can see that the original words are often on the front positions. Also it seems that LTLM is more syntactically oriented, which confirms claims from (Levy and Goldberg, 2014; Padó and Lapata, 2007), but to draw such conclusions a deeper analysis is required. The properties of the model strongly depends on the number of distinct roles. We experimented with maximally 1000 roles. To catch the meaning of various words in natural language, more roles may be needed. However, with our current implementation, it was intractable to train LTLM with more roles in a reasonable time. Training 1000 roles LTLM took up to two weeks on a powerful computational unit.

7 Conclusion and Future Work

In this paper we introduced the Latent Tree Language Model. Our model discovers the latent tree structures hidden in natural text and uses them to predict the words in a sentence. Our experiments with English and Czech corpora showed dramatic improvements in the predictive ability compared with standalone Modified Kneser-Ney LM. Our Java implementation is available for research purposes at <https://github.com/brychcin/LTLM>.

It was beyond the scope of this paper to explicitly test the semantic and syntactic properties of the model. As the main direction for future work we plan to investigate these properties for example by comparison with human-assigned judgments. Also, we want to test our model in different NLP tasks (e.g. speech recognition, machine translation, etc.).

We think that the role-by-role distribution should depend on the distance between the parent and the child, but our preliminary experiments were not met with success. We plan to elaborate on this assumption. Another idea we want to explore is to use different distributions as a prior to multinomials. For example, Blei and Lafferty (2006) showed that the logistic-normal distribution works well for topic modeling because it captures the correlations between topics. The same idea might work for roles.

Acknowledgments

This publication was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme “Projects of Large Research, Development, and Innovations Infrastructures”. Lastly, we would like to thank the anonymous reviewers for their insightful feedback.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic lan-

- guage model. *Journal of Machine Learning Research*, 3:1137–1155, March.
- David M. Blei and John D. Lafferty. 2006. Correlated topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120. MIT Press.
- Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. 2012. The joy of parallelism with czeng 1.0. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.
- Tomáš Brychcín and Miloslav Konopík. 2014. Semantic spaces for improving language modeling. *Computer Speech & Language*, 28(1):192–209.
- Tomáš Brychcín and Miloslav Konopík. 2015. Latent semantics in language models. *Computer Speech & Language*, 33(1):88–108.
- Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University.
- Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2009. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems 21*, pages 1–8.
- Arthur P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38.
- Koen Deschacht, Jan De Belder, and Marie-Francine Moens. 2012. The latent words language model. *Computer Speech & Language*, 26(5):384–409.
- Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.
- William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June. Association for Computational Linguistics.
- Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127.
- Martin Lange and Hans Leiß. 2009. To cnf or not to cnf? an efficient yet presentable version of the cyk algorithm. *Informatica Didactica*, 8.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.
- David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT’05*, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, volume 2010, pages 1045–1048. International Speech Communication Association.
- Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5528–5531, Prague Congress Center, Prague, Czech Republic.
- Thomas P. Minka. 2003. Estimating a dirichlet distribution. Technical report.
- Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, June.
- Martin Popel and David Mareček. 2010. Perplexity of n-gram and dependency language models. In *Proceedings of the 13th International Conference on Text, Speech and Dialogue, TSD’10*, pages 173–180, Berlin, Heidelberg. Springer-Verlag.
- Valentin I. Spitzkovsky, Hiyani Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010. Viterbi training

improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden, July. Association for Computational Linguistics.

Valentin I. Spitzkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Edward W. D. Whittaker and Philip C. Woodland. 2003. Language modelling for russian and english using words and classes. *Computer Speech & Language*, 17(1):87–104.